VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ

DEPARTMENT OF INFORMATION SYSTEMS

APLIKACE PRO POROVNÁNÍ VERZÍ SOUBORŮ SAP CRYSTAL REPORTS

AN APPLICATION FOR COMPARING SAP CRYSTAL REPORTS FILE VERSIONS

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — **JIŘÍ BUŠ**

AUTHOR

VEDOUCÍ PRÁCE — **RNDr. MAREK RYCHLÝ, Ph.D.**

SUPERVISOR

BRNO 2018

Zadání bakalářské práce/21226/2017/xbusji01

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2017/2018

Zadání bakalářské práce

Řešitel: **Buš Jiří**
Obor: Informační technologie
Téma: **Aplikace pro porovnání verzí souborů SAP Crystal Reports**
An Application for Comparing SAP Crystal Reports File Versions
Kategorie: Informační systémy

Pokyny:

1. Prostudujte objektový model tiskových šablon SAP Crystal Reports. Seznamte se s možnými reprezentacemi tohoto modelu a porovnejte také s jinými existujícími nástroji pro tiskové šablony a jejich reprezentacemi.
2. Navrhněte textový strukturovaný formát pro bezeztrátovou reprezentaci tiskových šablon SAP Crystal Reports. Formát by měl být vhodný pro porovnání různých šablon či jejich verzí. S využitím uvedeného formátu navrhněte aplikaci pro porovnání (diff) tiskových šablon a interaktivní procházení změn.
3. Po konzultaci s vedoucím navrženou aplikaci implementujte. Výsledek důkladně otestujte.
4. Zdokumentujte výsledky a navrhněte možná rozšíření.

Literatura:

- Cynthia Moore. Crystal Reports 2011 for Developers. 1st Edition. Cengage Learning PTR, 2011. ISBN 978-1435457966
- Crystal Reports Basic Tutorial. Tutorials Point. [https://www.tutorialspoint.com/crystal_reports/]
- George Peck. Crystal Reports XI: The Complete Reference (Osborne Complete Reference Series). McGraw-Hill Education, 2005. ISBN 978-0072262469. [http://crystalbook.com/]

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Rychlý Marek, RNDr., Ph.D.**, UIFS FIT VUT
Datum zadání: 1. listopadu 2017
Datum odevzdání: 16. května 2018

L.S.

doc. Dr. Ing. Dušan Kolář
vedoucí ústavu

Abstrakt

Tato práce se zabývá problematikou porovnání dvou souborů se šablonou vytvořených v aplikaci Crystal Reports. Tyto soubory jsou binární, a proto jsou běžnými prostředky neporovnatelné. Z toho důvodu je nejprve převedeme do XML struktury a porovnávací aplikaci spouštíme až na tyto XML soubory.

Abstract

This thesis solves the problém of comparing two different versions of reports created in Crystal Reports application. These binary files are basicly incomparable, so we transfer them to XML structure and then compare these XML files.

Klíčová slova

Crystal Reports, šablona, XML, porovnání rpt souborů, report, SAP, TortoiseSVN

Keywords

Crystal Reports, template, XML, compare of rpt files, report, SAP, TortoiseSVN

Citace

BUŠ, Jiří. Aplikace pro porovnání verzí souborů SAP Crystal Reports. Brno, 2018. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce RNDr. Marek Rychlý, Ph.D.

Aplikace pro porovnání verzí souborů SAP Crystal Reports

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením RNDr. Marka Rychlého. Další informace mi poskytl Ing. Petr Sládek. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Jiří Buš
16.5.2018

Poděkování

Chtěl bych poděkovat svému vedoucímu Marku Rychlému za odborné vedení, rady a pomoc při řešení této práce, dále pak všem kolegům z MP Orgy, rodině a přátelům, kteří mě podporovali v této bakalářské práci.

Obsah

1 Úvod

Tato práce se zabývá problematikou porovnávání dvou souborů tiskových šablon vytvořených v aplikaci Crystal Reports od společnosti SAP. Tyto soubory s příponou .rpt jsou jako binární soubory v běžných porovnávacích softwarech (diff) neporovnatelné, respektive výstup takového běžného porovnání je nečitelný. Nelze tak zjistit například změny, které se v souboru provedli, jelikož binární data se mohou změnit i pouhým otevřením souboru a jeho znovu uložením i když se bude jednat o tutéž šablonu, tak mohou být binární data rozdílná. Řešení toho problému si podrobně probereme v této bakalářské práci. Základní myšlenka je bezztrátový převod všech prvků souboru šablony do XML reprezentace a následné porovnání těchto lépe čitelných XML souborů. Pro převod do XML využiji rozhraní pro práci s tiskovými šablonami v jazyce C# vytvořené přímo společností SAP a dostupné pro vývojáře zdarma.

Na rozhraní existuje více objektových modelů, které definují, jakým způsobem je možné se souborem s šablonou pracovat. V této práci si detailněji probereme dva z nich. První z nich je „ReportDocument object model“ tento model lze využít pouze pro čtení souboru se šablonami a neobsahuje veškerou funkcionalitu, a tak není možné pomocí něj navrhnout aplikaci pro bezztrátový převod do XML, avšak jej lze využít pro jednodušší převod méně složitých reportů, které nevyužívají uživatelské formule. Druhý model je „ReportClientDocument object model“ tento model je naopak možné využít i pro zápis do souboru s šablonou, a tudíž se nabízí možné rozšíření této práce o možnost sjednotit dvě změny, pokud nemají vzájemný konflikt, do jedné a následně provést opačný převod z XML do rpt souboru. Dále tento druhý model popisuje veškeré vlastnosti, které lze v souboru se šablonou nastavit a je tedy vhodný pro bezztrátový převod do XML. Objektové modely by se neměly kombinovat z důvodu, že se částečně překrývají.

Pro porovnání XML souborů lze následně využít jakýkoliv program typu diff. Tím lze tedy dosáhnout širokou škálu využití a každý uživatel si může vybrat nástroj na porovnání dle vlastních preferencí. Jako příklad využití si uvedeme možnost napojení vytvořené aplikace do grafického rozhraní verzovacího systému subversion TortoiseSVN.

Dále si v této práci rozebereme několik již existujících nástrojů, které řeší problematiku porovnávání dvou souborů se šablonou, jako například ReportMiner, RPTtoXML, R-Tag. Většina z dostupných nástrojů je placených, ale také pak poskytují větší možnosti pro práci s reportem nebo jeho analýzu či vytváření dokumentace. Pro ověření správnosti řešení a představení práce s vytvořeným nástrojem si nakonec ukážeme výsledky z testů a podrobně zde popíši dosažené závěry a celkové zhodnocení přínosů práce.

2 Využívané nástroje

V průběhu řešení jsem využíval hlavně aplikaci Crystal Reports a její programové rozhraní v .NET pro Visual Studio a současně tedy i Visual Studio, jelikož je výsledná aplikace napsána v jazyce C#. Dále pro porovnání změn aplikaci KDiff a nástroj diff v aplikaci TortoiseSVN.

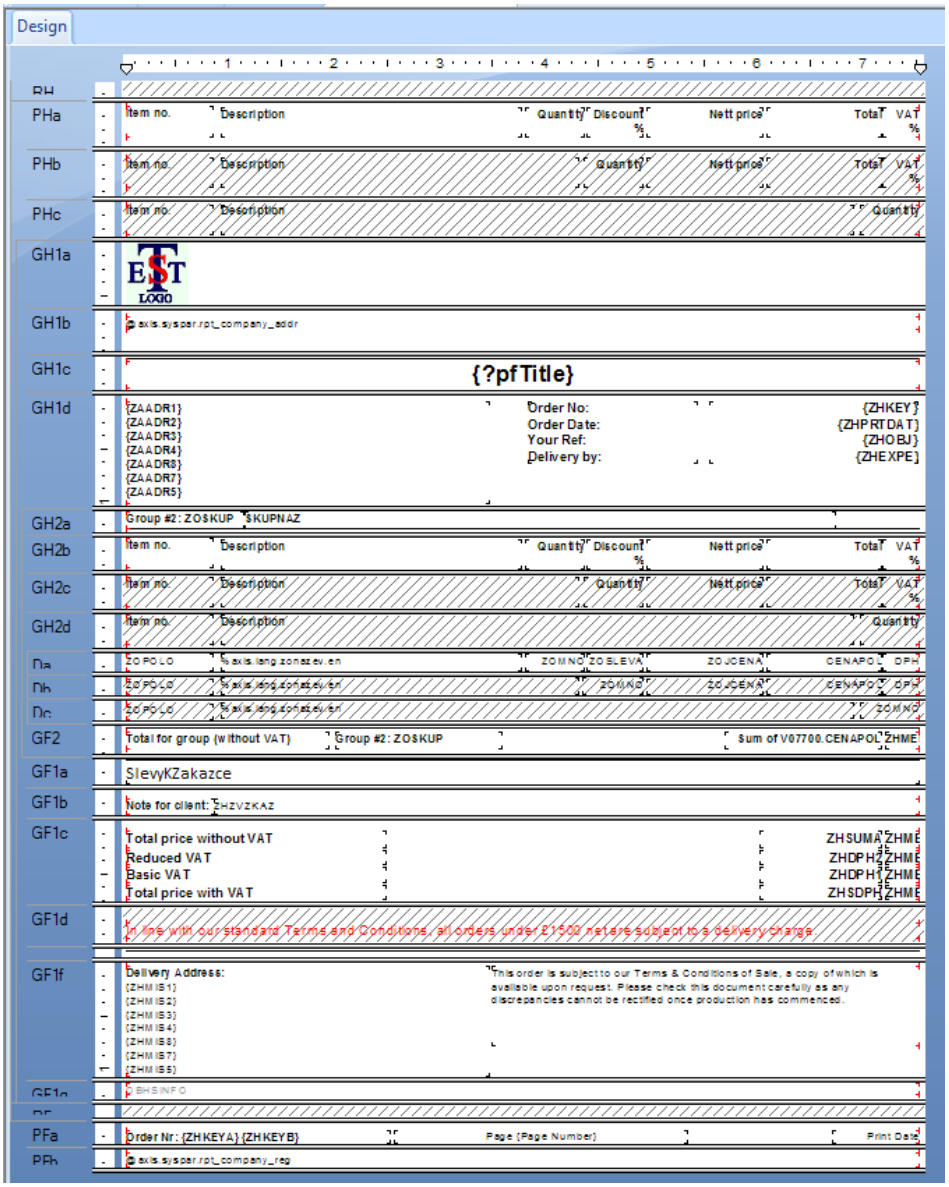
2.1 Crystal Reports

Crystal Reports je aplikace vyvíjena společností SAP a je to mocný nástroj pro vytváření tiskových výstupů z různých datových zdrojů a lze tedy jednoduše pomocí něj vytvořit šablonu, která se naplní daty z databáze a je poté možné ji vytisknout nebo případně exportovat do jiného cílového formátu [1]. Poskytuje široké možnosti nastavení formátu textu či různé nastavení vzhledu celé stránky. Lze taktéž využít pro export do excelové tabulky, dokumentu pro MS Word, stránky v HTML, dokumentu ve formátu PDF nebo také formáty RTF či CSV. Navíc obsahuje i vnitřní logiku, kde je možné nastavovat uživatelské formule, které se před vytisknutím vyhodnotí a lze tak předzpracovat data z databáze do jiné podoby. Do formulí je možné zapisovat i některé z přednastavených funkcí, či dokonce lze vytvářet funkce vlastní.

Na obrázku [Obrázek 2.1] je vidět, jak může vypadat editor šablon v reálně využitelném formátu. Lze zde vidět, že je rozdělen do několika oblastí, které jsou v levém sloupci popsány zkratkami.

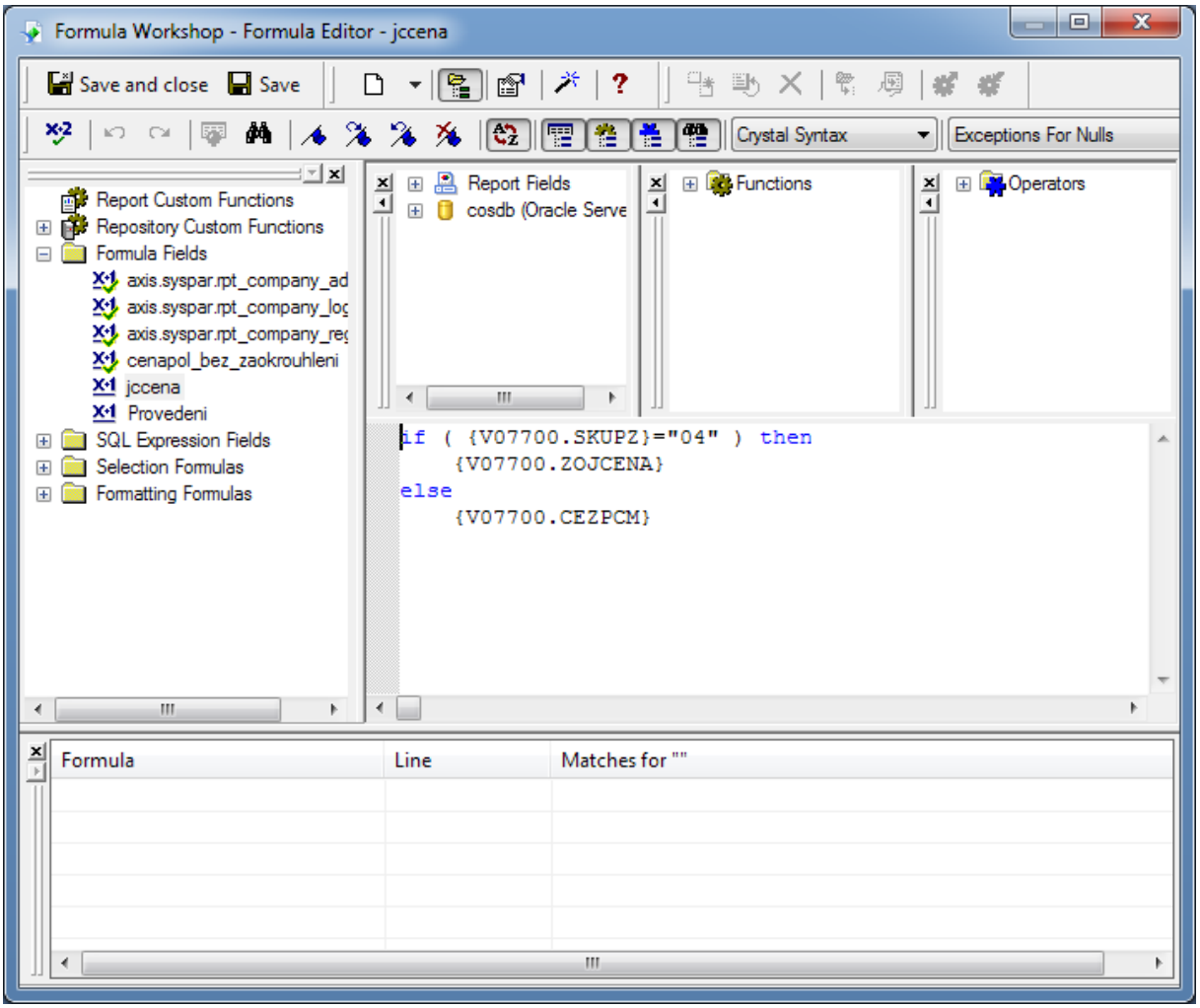
- Report header (RH) – záhlaví dokumentu ve výsledném souboru se zobrazí jen jednou na začátku dokumentu
- Page header (PH) – záhlaví stránky se zobrazí na každé nové stránce výsledného souboru
- Group header (GH) – hlavička pro každé seskupení
- Details (D) – jednotlivé položky
- Group footer (GF) – patička každého seskupení
- Report footer (RF) – zápatí celého výsledného souboru
- Page footer (PF) – zápatí každé stránky vytvořené ve výsledném souboru

Každá z těchto oblastí může mít libovolný počet sekcí, které jsou označeny písmeny abecedy. Oblast Group header a Group footer jsou navíc rozděleny číslem podle úrovně seskupení.



Obrázek 2.1 Ukázka editoru šablon reportu

Každé jednotlivé políčko v šabloně, může mít vlastní nastavení formátu a také jiný zdroj dat. Políčka mohou obsahovat přímo předepsaný text, který se v případě vytváření výstupního souboru již nemění anebo obsahují proměnné, které se nejprve vyhodnotí a zobrazí se poté jejich vyhodnocená hodnota. Políčka ve složených závorkách jsou dotahována z databáze. Pokud je na začátku identifikátoru ve složených závorkách otazník, jde o parametr, který se nastavuje poté co uživatel požádá o vytvoření výsledného souboru. Pokud identifikátor začíná procentem jde o výraz v jazyce SQL a pokud zavináčem, tak jde o formuli.



Obrázek 2.2 Editor formulí

Formule lze nastavit pomocí vlastního programovacího jazyka, který obsahuje spoustu vestavěných funkcí. Lze zde využít i různé hodnoty dotažené z databáze nebo vypočítané za pomoci operátorů, jak je vidět například na obrázku [Obrázek 2.2].

Je zde možné zadat i políčko, které v sobě obsahuje další celou šablonu a nazývá se subreport. V subreportu již další subreport být nemůže a lze takto dosáhnout pouze jedné úrovně zanoření. Jinak jsou možnosti subreportu téměř totožné s možnostmi hlavního reportu. Navíc je také definována vazba mezi reportem a jeho subreporty s možností sdílet některá políčka [6].

2.1.1 Verze

Pro řešení jsem využíval verzi 14.0.x z roku 2011 a pro následné otestování jsem využil i některé šablony vytvořené v Crystal Reports XI z roku 2004. Neměl by však být problém i s novějšími verzemi. Poslední aktuální verze je 14.2.x z roku 2016.

2.2 Aplikace pro porovnání změn

Aplikací pro porovnávání změn ve dvou souborech existuje spousty. Základním principem všech takovýchto nástrojů je, že na vstupu jsou dva soubory stejného typu a výstupem je textové nebo grafické porovnaní těchto dvou vstupních souborů a vyznačení změněných částí a případně poskytuje další úpravy porovnávaných souborů. Porovnávací programy s textovým výstupem vypíší většinou pouze řádky, které se ve zdrojových souborech liší. Naopak při grafickém zobrazení jsou dostupné oba soubory zobrazené vedle sebe a jednotlivé rozdíly lze snadno procházet navigačními šipkami a jsou barevně rozlišené.

Z dostupných aplikací pro porovnání změn jsem využil aplikaci KDiff, která je zdarma dostupná ke stažení. Dále jsem taktéž využil zabudovaný nástroj v aplikaci TortoiseSVN, který je také vhodný pro porovnávání XML souborů a poskytuje možnosti pro snadnou navigaci a procházení těchto změn.

TortoiseSVN je grafické rozhraní pro SVN (Subversion). Je vhodný pro jednodušší ovládání verzovacího software a veškeré operace lze provádět pomocí dialogů doplněných do kontextového menu Windows [4].

2.3 Způsob zápisu dat

Nutnou součástí návrhu výsledné aplikace pro porovnávání dvou šablon byl výběr vhodného textového formátu, který musel splňovat podmínky snadné čitelnosti pro uživatele a možnosti strukturovat text do stromové struktury, aby bylo možné do takovéhoto formátu převést objektový model rozhraní Crystal Reports. Z dostupných strukturovaných textových formátů jsem se rozhodoval mezi XML, JSON a YAML.

2.3.1 XML

XML je značkovací jazyk, ve kterém je možné zapisovat vlastní názvy značek i jejich atributů přesně podle potřeb a způsobu jeho použití. Byl odvozen z jazyka SGML a vyvinut k publikování rozsáhlých elektronických dat. V současnosti je široce využíván pro výměnu elektronických dat mezi různými systémy nejen přes internet [2]. Aby dokument v jazyce XML byl správně strukturovaný (*well-formed*) je potřeba aby splňoval následující podmínky, které definuje W3C.

- Musí obsahovat jednu nebo více značek, přičemž právě jedna je kořenová značka (*root element*) tedy celý dokument musí být uzavřen pomoc otevírací a ukončovací značky právě tohoto kořenového elementu.
- Všechny elementy musí být párové, tedy mít počáteční a ukončovací značku s výjimkou prázdných elementů u těch stačí pokud je jedna značka ukončena znaky />.
- Všechny atributy elementů musí být uzavřeny v uvozovkách nebo apostrofech.

2.3.2 JSON

JSON je založen na programovacím jazyku Java script. Jedná se o textový formát zápisu dat takovým způsobem, aby byl snadno zapisovatelný i čitelný člověkem, ale současně i generovatelný a čtený pomocí počítačových programů. Dvě základní struktury, které využívá JSON jsou objekt a pole. Objekt je uzavřený ve složených závorkách a obsahuje dvojice jméno a hodnota oddělené čárkami. Pole je seznam hodnot v hranatých závorkách. Nevýhodou JSON je nemožnost definovat znakovou sadu, ve které jsou textová data uložena [7].

Na jednu stranu poskytuje zdánlivě kratší zápis díky absenci značek, které jsou v XML, ale dle mého názoru není tolik přehledný pro hledání chyb jako XML a stejný počet zapsaných dat zabírá v textovém formátu JSON téměř dvojnásobek místa na disku než v případě zápisu pomocí XML.

2.3.3 YAML

Podobně jako v JSON tak i YAML, jak jeho název YAML Ain't Markup Language napovídá, neobsahuje značky [9]. Mezi jeho základní prvky patří:

- Sekvence – prostý seznam prvků
- Mapa – podobně jako v JSON objekt (dvojice klíč: hodnota)
- Dokument – logický celek, který se může v YAML souboru objevit vícekrát se odděluje pomocí tří spojovníku na samostatném řádku

Dále lze tyto základní prvky vzájemně kombinovat. Stromovou strukturu tvoří jen pomocí odsazování řádků, díky čemu mi opět nepřijde příliš vhodný pro porovnávání dvou soborů, jelikož ne všechny porovnávací nástroje berou ohled na bílé znaky. Zápis pomocí YAML je, co se týče velikosti souboru, pro strukturu dat vytvořenou v této práci, někde mezi XML a JSON.

2.3.4 Výsledek

Pro řešení této práce jsem si vybral XML, jelikož lze vhodně využít pro popis struktury jednotlivých objektů programového rozhraní Crystal Reports a oproti ostatním možnostem mi přijde nejpřehlednější i pro rozsáhlé textové soubory. Při testování jsem z XML souboru, který měl tisíc řádků, po převedení do formátu JSON, dostal šestkrát tolik řádků a nebylo snadné při změně atributu u některého prvku v porovnání souborů pak nalézt, ke kterému prvku daný atribut náleží a tím pádem nešlo snadno zjistit, jaká změna v souboru se šablonou byla provedena.

V případě, že by XML reprezentace byla nevyhovující je možné rozšířit vlastnosti programu a provést převod XML do JSON, YAML či jiného formátu pomocí některého z mnoha dostupných nástrojů [8].

2.4 Programové rozhraní

Crystal Reports poskytuje zdarma rozhraní pro Visual Studio určené pro vývojáře [5]. Je podporováno od verze Visual Studia 2010. Poslední verzí tohoto rozhraní je Crystal Reports for Visual Studio 13.0.22, kterou jsem využil v tomto řešení. Pro možnost využívat řešení této práce je nutné mít toto rozhraní nainstalované. Na internetových stránkách Crystal Reports[1] je možné po zadání e-mailové adresy a státu dostat zdarma instalační soubor tohoto rozhraní.

Instalace rozhraní vyžaduje mít nainstalované Visual Studio. Po nainstalování je možné v uživatelském rozhraní Visual Studia prohlížet a upravovat soubory se šablonou podobně jako v aplikaci Crystal Reports.

Dále toto rozhraní obsahuje čtyři objektové modely, které není vhodné kombinovat. V mém řešení jsem využil dva z objektových modelů ovšem odděleně, každý model v jiné třídě, tím tedy k jejich kombinaci nedochází. Dva z modelů využívám v mém řešení hlavně kvůli možnosti následného otestování a určení, který z modelů je pro řešení vhodnější, případně pro jaký způsob využití je, který model více vhodný.

2.4.1 CrystalReportViewer object model

Nejjednodušší objektový model. Jeho třídy pouze umožňují práci s výsledným zobrazením již vyplněné šablony daty z databáze. Jmenné prostory CrystalDecisions.Windows.Forms a CrystalDecisions.Web. Lze využít pouze pro zobrazení již vyplněné šablony v aplikaci Windows nebo webové aplikaci.

2.4.2 ReportDocument object model

Širší objektový model, který obsahuje základní prvky, kterými se dají popsat šablony. Lze využít pro načtení souboru se šablonou a následně jej pak exportovat nebo vytisknout. Nelze pomocí něj však šablonu upravovat a neobsahuje ani popis jednotlivých uživatelsky definovatelných formulí uvnitř šablony. Je dostupný ve jmenném prostoru CrystalDecisions.CrystalReports.Engine.

2.4.3 ReportClientDocument object model

Tento objektový model poskytuje třídy, které úplně popisují veškeré možné atributy a funkce, které lze využívat v souboru se šablonou. Navíc poskytuje metody, pomocí kterých lze soubor se šablonami vytvářet či editovat. Je rozšířen po více jmenných prostorech, každý z nich začíná na CrystalDecisions.ReportAppServer. Diagram tříd tohoto a předchozího modelu je dostupný na přiloženém paměťovém médiu.

[1] https://www.crystalreports.com/crystal-reports-visual-studio/

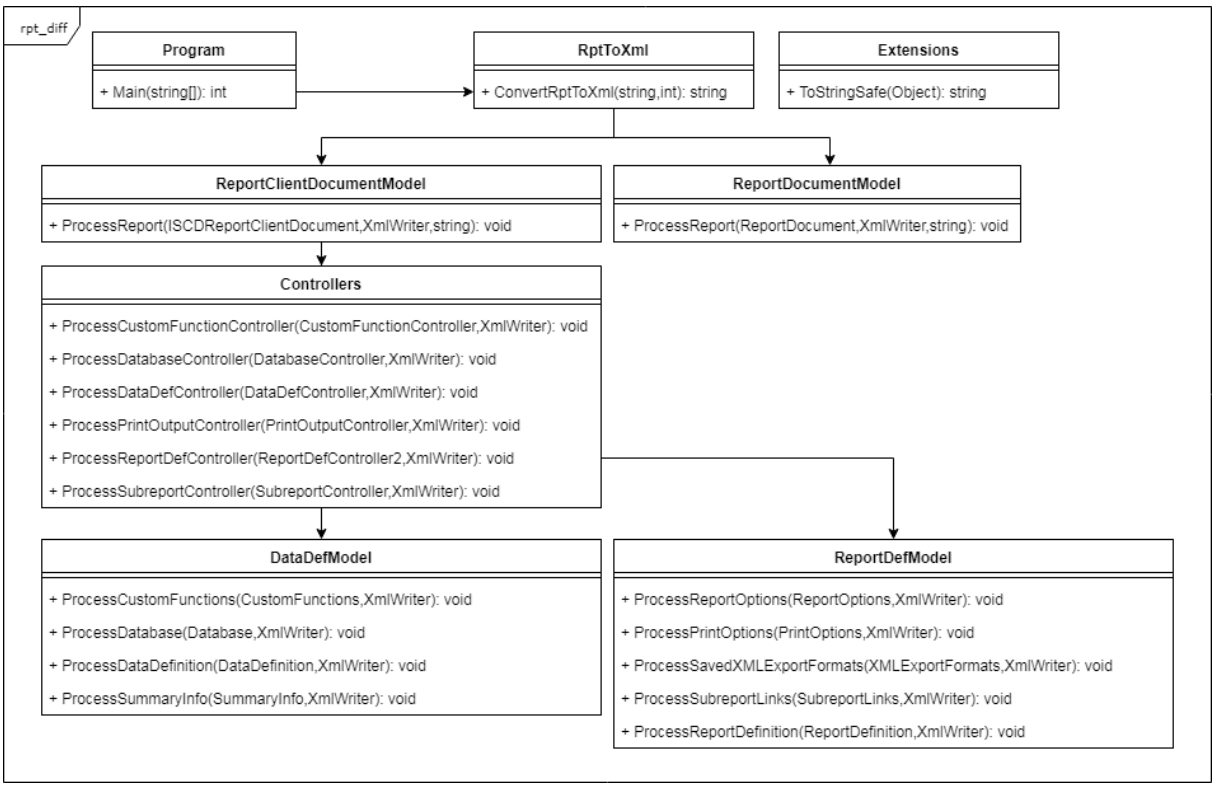
2.4.4 InfoObject object model

Tento model byl navržen pro spolupráci s podnikovými funkcemi Crystal Reports Server nebo BusinessObjects Enterprise. Jmenný prostor CrystalDecisions.Enterprise [3].

3 Převod do XML

Klíčovou součástí této práce je převod binárního souboru se šablonou do XML reprezentace. Tento přenos je bezztrátový a XML značky jsou pojmenovány podle jednotlivých tříd a jejich atributů, které jsou dostupné z rozhraní. Převod je realizován pomocí dvou modelů, ze kterých je možné si při spuštění aplikace vybrat zadáním parametru. První možnou volbou je ReportDocument model, který je jednodušší a nepokrývá veškeré atributy souboru se šablonou. Tento model je vhodné použít pro rychlejší převod a pro jednoduché šablony, ve kterých se nepoužívají uživatelské formule. Druhý možný převod, který podporuje vytvořená aplikace je podle modelu ReportClientDocument. Tento převod je více časově náročný, ale naproti tomu obsahuje veškeré dostupné informace o souboru se šablonou. Drobnou nevýhodou tohoto přístupu je, že pokud se některé formule v souboru se šablonou nevyužívají jsou pak ve výsledném XML prázdné značky.

Pro vytváření XML souboru je použita třída `XmlTextWriter` ze standartní knihovny System.Xml.



Obrázek 3.1 UML diagram tříd řešení obsahující jen veřejné metody

3.1 Popis implementace

Na obrázku [Obrázek 3.1] je vidět UML diagram všech vytvořených tříd, které zajišťují převod binárního souboru se šablonou do XML reprezentace. Tento diagram obsahuje pouze veřejné (*public*) metody tříd. Kompletní diagram včetně soukromých (*private*) metod je dostupný v příloze.

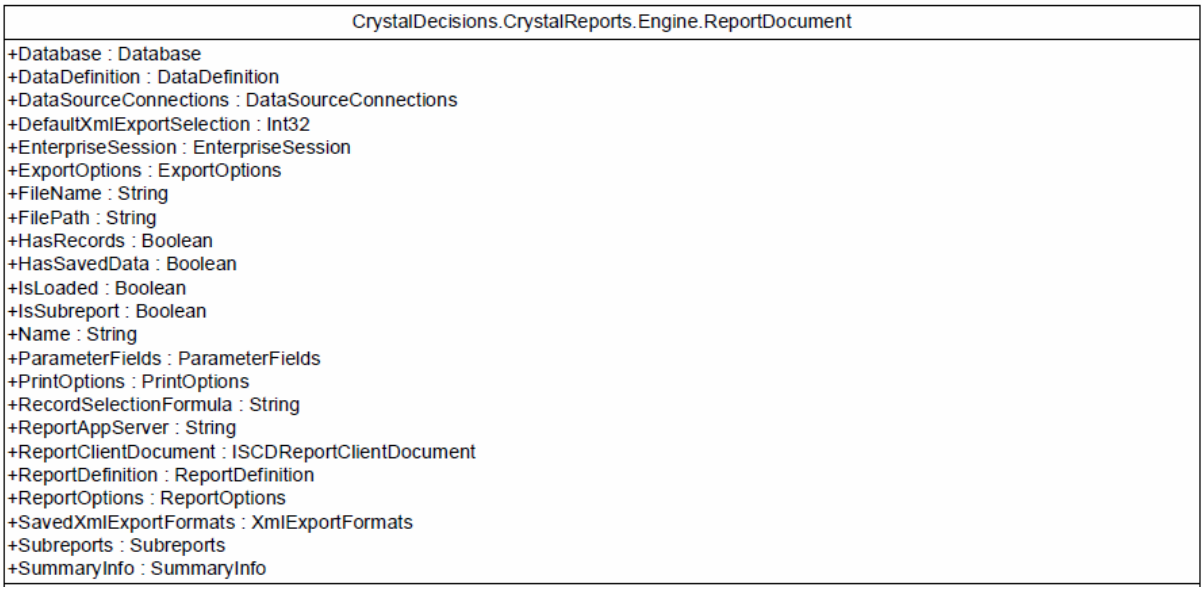
Na třídu `Extensions` se odkazuje ze všech míst, kde se využívá její metoda pro převod objektu do řetězce znaků. V diagramu pro přehlednost nejsou tyto vazby vyznačeny. Dále zde nejsou vyznačeny přímé vazby ze třídy `ReportClientDocumentModel` do tříd `DataDefModel` a `ReportDefModel`.

Po spuštěni programu se ve třídě Program podle zadaných parametrů otestuje jejich správnost a spustí se převod souboru se šablonou do XML reprezentace a následně se na XML soubory zavolá aplikace pro jejich porovnání. Hlavní vstupní třidou převodu je třída `RptToXml`, která obsahuje pouze jednu metodu `ConvertRptToXml` s jedním parametrem, který specifikuje cestu k souboru se šablonou a druhým parametrem se určí, který z modelů pro převod použít. Tato metoda otevře soubor se šablonou jako dočasnou kopii a není tedy možné soubor zadaný parametrem nijak změnit. Dále vytvoří ve stejném umístění na disku jako se nachází soubor se šablonou prázdný soubor XML se stejným názvem jako je název šablony, který se bude postupně dále naplňovat.

3.2 ReportDocument

V případě, že je parametr určující model nastavený na 0, tak se pro převod použije ReportDocument model. Tento model je implementován jednou třídou `ReportDocumentModel`. Vstupní metodou této třídy je metoda `ProcessReport`, které je nutné předat otevřený soubor reportu a odkaz na XML.

Z této metody se volají další již privátní metody, které mají vždy za úkol zapsat do XML začátek značky s názvem objektu, který se právě zpracovává. Poté atributy daného objektu a pokud má objekt nějaké podobjekty, tak zpracuje tyto podobjekty, a nakonec zapíše ukončovací značku.



Obrázek 3.2 Hlavní objekt popisující celý soubor se šablonou v modelu ReportDocument

Na obrázku [Obrázek 3.2] je vidět UML diagram pro třídu `ReportDocument`, která je dostupná na rozhraní v tomto objektovém modelu a obsahuje všechny prvky, které lze z tohoto modelu získat.

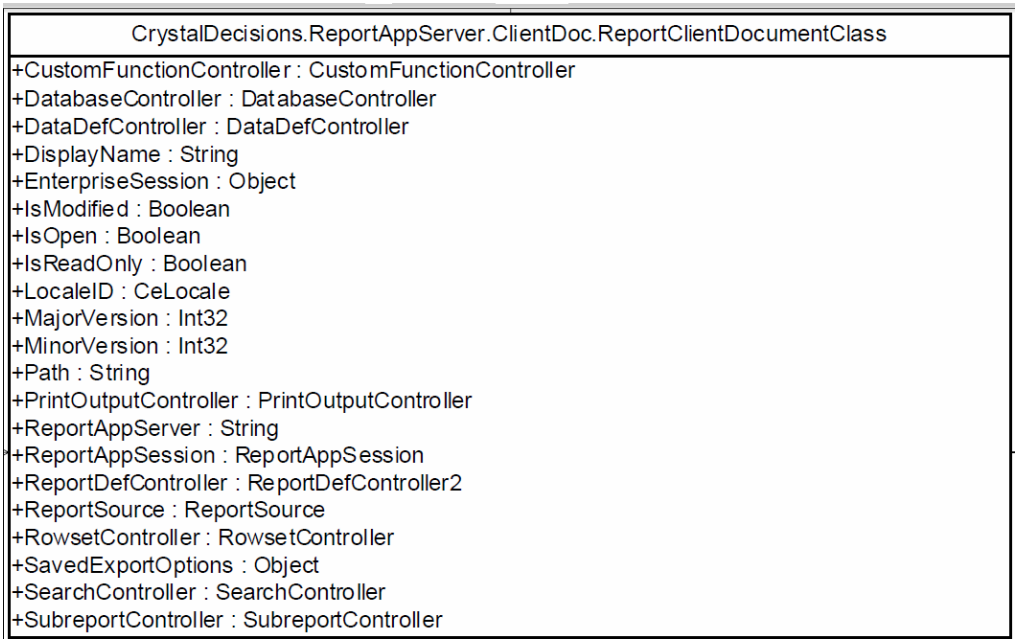
Veškeré hodnoty atributů se zapisují v podobě řetězce a pro tuto potřebu jsem vytvořil pomocnou metodu ve třídě Extensions `ToStringSafe`, která převede objekt na řetězec a pokud je objekt *null*, tak vrací prázdný řetězec.

3.3 ReportClientDocument

Pokud metodě `ConvertRptToXml` zadáme jako druhý parametr číslo 1, tak se převod bude provádět pomocí kompletního objektového modelu ReportClientDocument. Zavolá se tedy opět metoda `ProcessReport`, ale tentokrát z třídy `ReportClientDocumentModel`. Podobně jako v předchozím modelu zapisuje pro každou třídu z rozhraní samostatnou XML značku a do značky její atributy. V případě, že objekt obsahuje nějaké podobjekty, tak jsou značky zanořené do další úrovně.

Vzhledem k tomu, že rozhraní pro tento model je rozděleno do několika jmenných prostorů tak jsou podle nich rozděleny třídy převodu. Třída `Controllers` tedy zpracovává jmenný prostor CrystalDecisions.ReportAppServer.Controllers, třída `DataDefModel` prochází jmenný prostor CrystalDecisions.ReportAppServer.DataDefModel a třída `ReportDefModel` pracuje se jmenným prostorem CrystalDecisions.ReportAppServer.ReportDefModel.

3.3.1 Třída ReportClientDocumentModel



Obrázek 3.3 Hlavní objekt popisující celý soubor se šablonou v modelu ReportClientDocument

Z hlavní třídy rozhraní v tomto modelu viz. [Obrázek 3.3], jsou dostupné tyto odkazy na třídy z jmenného prostoru Controllers:

- CustomFunctionController – popisuje uživatelsky definovatelné funkce
- DatabaseControler – připojené databázové objekty jako jsou tabulky a jejich sloupce a případně vazby mezi nimi
- DataDefControler – ostatní políčka, která nejsou z databáze
- PrintOutputController – nastavení tisku, rozvržení stránky a podobně
- ReportDefController – grafické nastavení jako výsledný vzhled stránky a formát jednotlivých polí
- SubreportController – další vnořený soubor se šablonou

Dále je zde dostupný odkaz do jmenného prostoru DataDefModel:

- SummaryInfo – souhrnné informace jako autor, titulek a podobné

A nakonec i do jmenného prostoru ReportDefModel:

- ReportOptions – nastavení souboru se šablonou

Atributy jednoduchého typu jako String, Boolean nebo Int jsou přímo zapsány do XML značky jako její atributy.

Ostatní atributy této třídy, které jsem nezmínil nejsou pro účel porovnání důležité, protože svou funkci plní až při připojení reálných dat z databáze, a to v okamžiku porovnávání šablon není k dispozici a ani není potřeba, protože jde o porovnání nastavení šablon, a ne o porovnání obsahu databází.

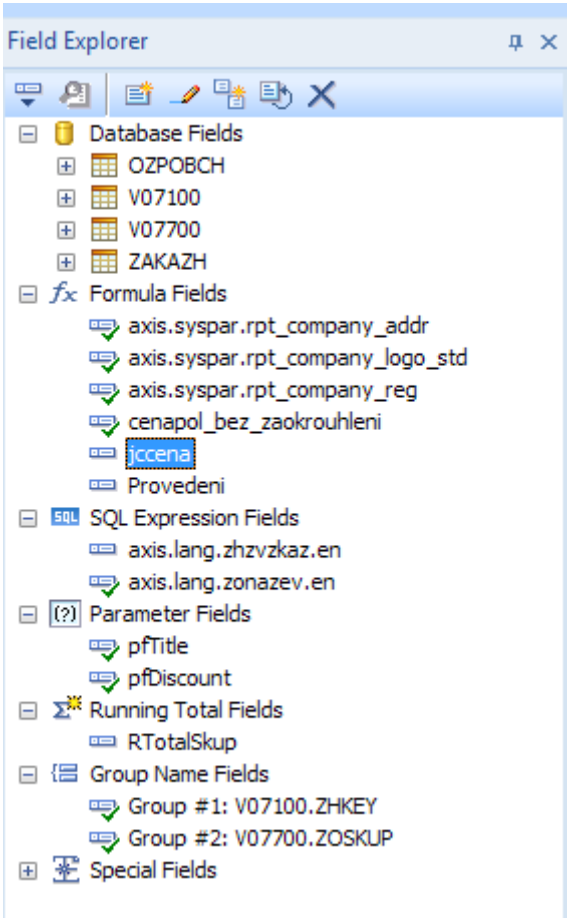
3.3.2 Třída Controllers

Většina metod v této třídě jen posouvá odkaz do další úrovně do tříd `DataDefModel` nebo `ReportDefModel` v závislosti na tom, zda jde o nastavení políček jakožto jejich datového obsahu nebo zda jde o nastavení vzhledu a formátu těchto polí přímo ve výsledném grafickém zobrazení případně informace o tisku nebo dalších informacích o souboru se šablonou. Proto také nezapisují do XML souboru značky, protože by pouze zvětšili zanoření o úroveň víš.

Výjimkou je metoda `ProcessSubreportController`, která zpracovává subreporty, kterých může být víc. Každý tento subreport je zpracováván přímo v této třídě a jelikož jeho vlastnosti jsou podobné celému souboru se šablonou, tak se odtud volají některé stejné metody jako se volaly z hlavní třídy (`DatabaseController, DataDefControler, ReportDefController` a `ReportOptions`). Navíc se zde zpracovávají atributy, které má nastavené a vazby mezi hlavním reportem a subreportem. Zpracování těchto vazeb probíhá ve třídě `ReportDefModel`.

3.3.3 Třída DataDefModel

Zde je prováděn převod všech prvků souboru se šablonou, který se přímo týká dat dotahovaných z datových zdrojů. Jde tedy především o zpracování připojené databáze, tabulek, jejich sloupců a vazeb mezi tabulkami. Jednou z dostupných metod zde je přepis uživatelských funkcí do XML veškeré dostupné informace jsou názvy takovýchto funkcí a jejich definice. Dále jsou zde zpracována všechna políčka, která jsou v souboru se šablonou nadefinována.



Obrázek 3.4 Seznam políček v souboru se šablonou

Na obrázku [Obrázek 3.4] je seznam možných políček, jak se zobrazuje v Crystal Reports. Převod do XML obsahuje všechny tyto políčka. Políčka z databáze jsou zpracovány poté co se zpracuje metoda `ProcessDatabase` nejprve se zde však zpracují tabulky a jejich vazby. V každé tabulce je pak seznam dostupných polí. Ostatní skupiny, které jsou vidět na obrázku se zpracovávají postupně pomocí metody `ProcessDataDefinition`. Drobný rozdíl je v tom, že Formula fields a SQL Expression fields jsou v převáděném XML spojené pod jednu značku FormulaFields. Tato metoda však zpracovává ještě další dostupné informace, které na obrázku nejsou vidět.

Mezi tyto informace patří:

- GroupFilter – filtr, který se uplatňuje pro výběr zobrazovaných dat do seskupení
- Groups – seskupení dat podobně jako v SQL dotazu klauzule GROUP BY
- RecordFilter – filtr, který se aplikuje na veškerá data předtím, než se zobrazí ve výsledné podobě
- SavedDataFilter – filtr na data, která lze uložit se souborem s šablonou
- Sorts – způsob řazení dat ve výsledném zobrazení obdoba v SQL je ORDER BY
- SummaryFields – sumarizační políčka například suma hodnoty v rámci seskupení

Zpracování všech políček probíhá pomocí společné metody `ProcessFields`, která pro každé jednotlivé pole volá `ProcessField`, která zpracuje nejprve informace dostupné pro všechny typy polí a následně podle toho o jaký druh pole se jedná zpracuje konkrétní informace pro tento druh.

Druhy polí:

- DB field – políčko z tabulky z databáze
- Formula field – uživatelsky nastavená formule
- Group Name field – pole, podle kterého je prováděno seskupení
- Parameter field – parametr, který se zadává před vytvořením výsledného souboru, může mít nastavené výchozí hodnoty
- Running Total field – políčka pro průběžné součty
- Special field – speciální pole jako aktuální čas, název reportu, počet stránek a podobně
- Summary field – sumarizační políčka

Další oblast, se kterou se zde pracuje jsou seskupení. V souboru se šablonou lze nastavit, aby podle libovolného políčka prováděl seskupení a v rámci toho seskupení lze nastavit řazení záznamů a případně další uživatelské formule popisující například, jak se má získat název tohoto seskupení. Toto vše zastřešuje metoda `ProcessGroups`.

Pak je zde ještě metoda `ProcessSorts`, která se stará o informace o řazení záznamů mimo seskupení. Jsou zde informace, jakým směrem a podle kterého pole se řazení provádí.

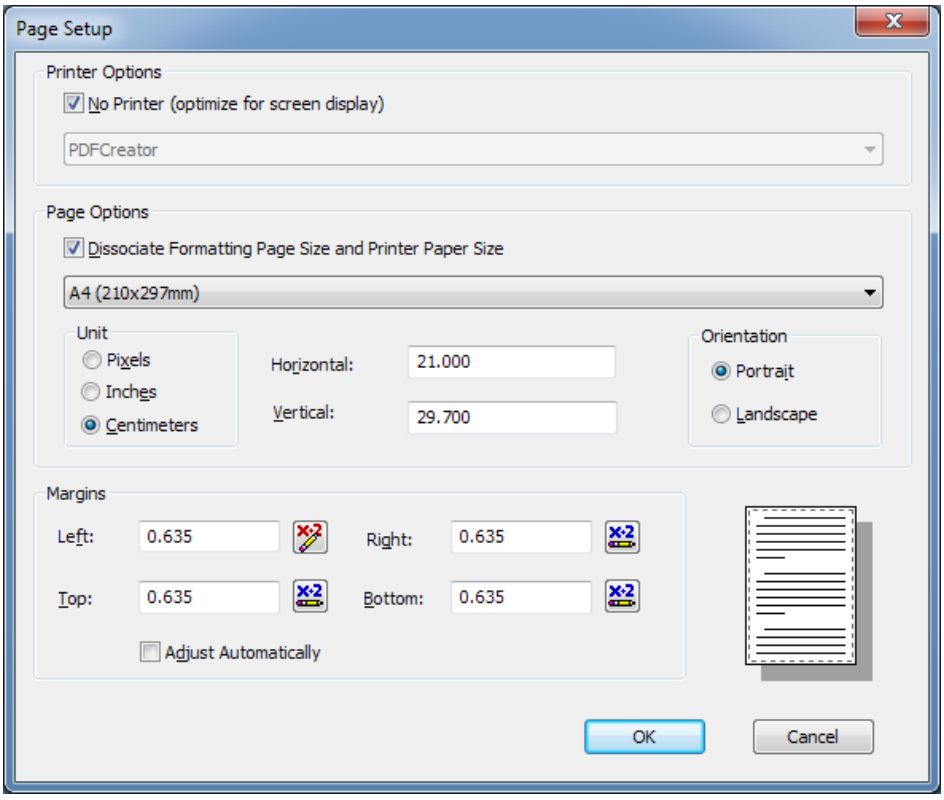
Nakonec je zde metoda `ProcessSummaryInfo`, která poskytuje informace o reportu jako například jeho autor, komentáře, název a podobné.

3.3.4 Třída ReportDefModel

Tato třída se stará o zpracování informací dostupných z grafického editoru jako je umístění jednotlivých polí v rámci jednotlivých sekcí a oblastí a nastavení těchto polí po stránce formátu. Dále zde je zpracováváno nastavení reportu a nastavení jeho tisku nebo exportu.

Nejprve se tedy podíváme na zpracování nastavení reportu, zde jsou dostupné informace z dialogu Report Options. Tato nastavení obsahuje každý soubor se šablonou, proto je metoda `PrecessReportOptions` volána přímo z hlavní třídy tohoto modelu `ReportClientDocumentModel`.

Dále se zpracovává nastavení tisku, které se při vytváření souboru se šablonou specifikuje v tomto dialogu.

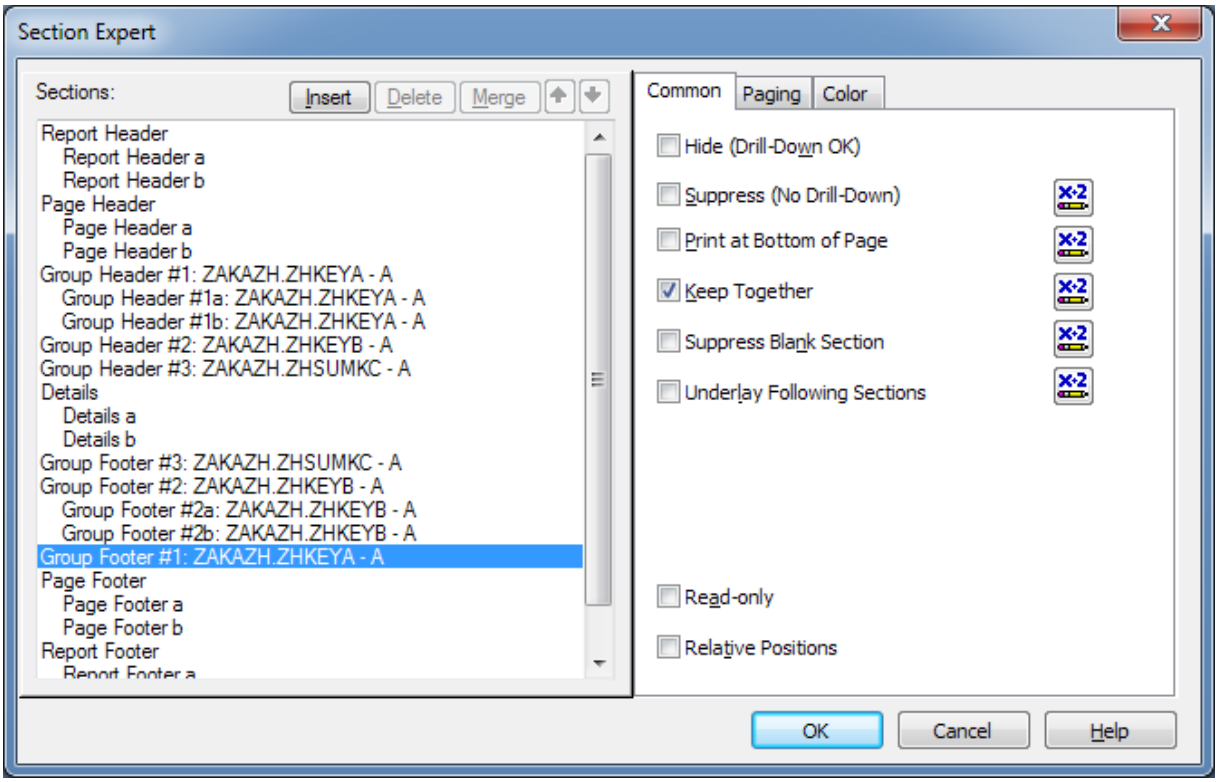


Obrázek 3.5 Dialog Page Setup

Na obrázku [Obrázek 3.5] je vidět v sekci Margins, že u jednotlivých hodnot s rozměry se nachází tlačítko  nebo  . To znamená, že pro tento rozměr lze nastavit uživatelskou formuli, která může obsahovat libovolnou funkci, která vrací číslo. V případě modrého x+2 znamená, že není formule vyplněná a pokud je tlačítko s červeným x+2 tak to znamená, že je uživatelská formule nastavená. Takovéto uživatelské formule je možné nastavit u mnoha různých vlastností, které Crystal Reports obsahuje.

Toto je právě hlavní rozdíl mezi modelem ReportDocument a ReportClientDocument. V Prvním z těchto modelů, takovéto uživatelsky zadávané formule nejsou dostupné, ale ve druhém z modelů ano. Ve výsledném XML jsou vždy tyto formule atributem značky XML, přičemž název tohoto atributu končí právě slovem „Formula“.

Velmi důležitou částí této třídy je metoda `ProcessReportDefinition`. Ta se stará o zpracování všech dostupných informací o oblastech reportu jejich jednotlivých sekcích a následně všech dostupných objektů, které byly naskládány do grafického editoru Crystal Reports. Vytvoří tedy takovou stromovou strukturu, pomocí které je pak možné snadno dohledat v jaké části se jaký objekt nachází a jaké jsou jeho nastavení fontu, ohraničení, velikosti a podobně, což je klíčové pro porovnávání změn dvou souborů se šablonou.



Obrázek 3.6 Nastavení oblastí

Každá oblast i její sekce, které jsme si popsali v kapitole 2.1 má mnoho nastavení i s možností nastavit uživatelské formule, jak je vidět v dialogu na obrázku [Obrázek 3.6]Obrázek 3.5. Veškeré tyto informace získávají metody `ProcessAreaFormat` a `ProcessSectionFormat`. Metoda `ProcessReportObject` vezme každý jednotlivý objekt, který je vidět v grafickém editoru Crystal Reports a podle jeho typu jej zpracuje.

Typy objektů jsem seskupil podle společných vlastností:

- BlobField – binární databázový objekt (většinou obrázek)
- Box – rámeček, který lze v editoru libovolně nakreslit
- FieldHeading – záhlaví provázáno s jiným objektem
- Field – datové pole dotahované z datového zdroje
- Line – čára, kterou lze v editoru nakreslit
- Subreport – objekt vnořeného souboru se šablonou
- Text – textové pole vyplněné libovonným textem, může obsahovat i datové pole

Všechny objekty mají část vlastností společných a část specifických. Společné vlastnosti jsou umístění, velikost a název, dále pak nastavení ohraničení a základní formátování objektu.

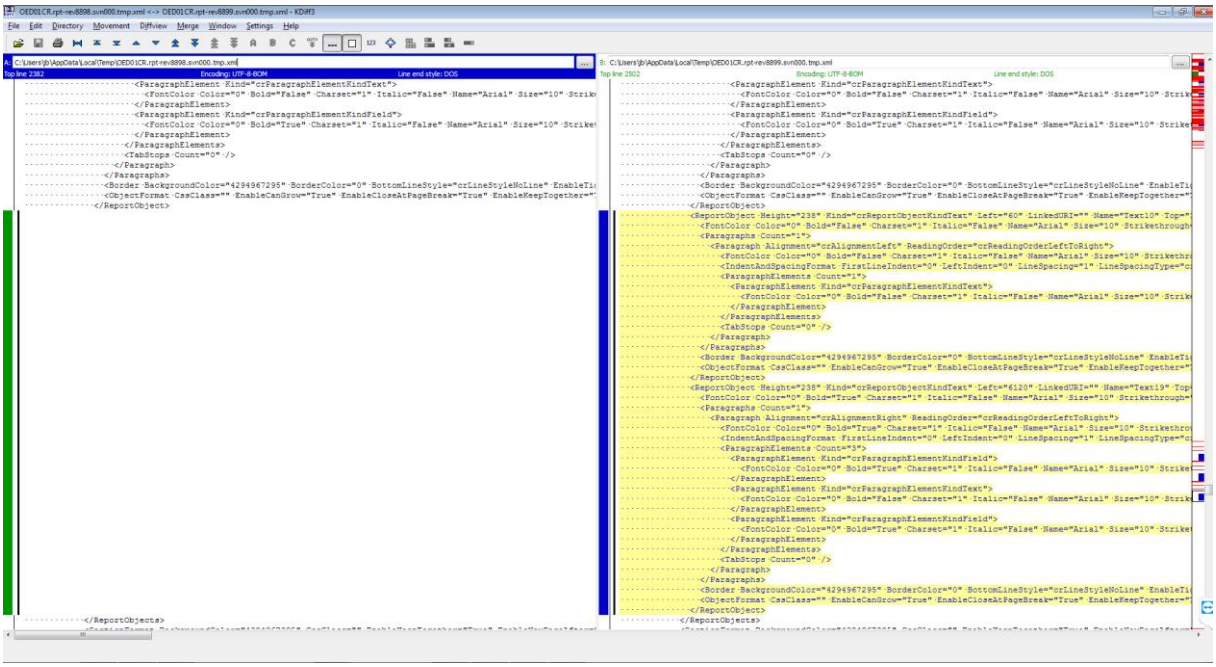
Každý objekt typu Field má navíc typ hodnoty, podle kterého obsahuje jiné vlastnosti, které lze nastavit.

Typy hodnot pro objekt typu Field:

- BooleanField – boolean hodnota pravda/nepravda
- DateField – datum bez časového údaje
- DateTimeField – datum včetně časového údaje
- NumberField – číslo (dále se stejným způsobem zpracovávají i typy CurrencyField, Int16sField, Int16uField, Int32sField, Int32uField, Int8sField, Int8uField)
- StringField – řetězec znaků
- TimeField – pouze časový údaj bez data

3.4 Porovnání rozdílů

Pro výsledné grafické porovnání je možné použít jakýkoliv nástroj pro porovnávání dvou XML souborů. V rámci testování jsem použil aplikaci KDiff, která je volně dostupná ke stažení. Aplikace obsahuje nástroje k procházení jednotlivých změněných částí a graficky je znázorňuje.



Obrázek 3.7 Grafické rozhraní aplikace KDiff

Na obrázku [Obrázek 3.7] je ukázka grafického rozhraní aplikace KDiff. Jedná se o jednoduché změny v datovém modelu a přidání šesti políček. Díky tomuto porovnání je velmi přehledně dohledatelné, co se v souboru se šablonou změnilo.

Při porovnání je důležité sledovat v jaké části XML souboru se rozdíly nachází. Sémantika výsledného XML souboru se může lehce lišit v závislosti na použitém objektovém modelu. Základní

kostra je však téměř totožná u obou modelů. Hlavní značka ReportDocument nebo ReportClientDocument obsahuje tyto další, které se dále rozvětvují:

- Database – zde je možné nalézt rozdíly v připojených datových zdrojích
- DataDefinition – pokud je zde rozdíl, tak se změnila definice některých polí
- PrintOptions – tato část informuje o změnách v nastavení tisku
- ReportDefinition – nejobsáhlejší část, která pokrývá veškeré změny nastavení grafického vzhledu reportu je přehledně rozdělena podle sekcí a oblastí
- ReportOptions – nastavení reportu
- SummaryInfo – souhrnné informace o autorovi a titulku reportu a podobně
- XMLExportFormats – nastavení formátu XML exportu dostupného z aplikace Crystal Reports

Atributy jednotlivých značek odpovídají svými názvy popisům v grafickém rozhraní Crystal Reports, proto by mělo být snadné rozpoznat, co který znamená. Některé informace jsou ve výsledném XML oproti grafickému rozhraní navíc. Například u všech políček je atribut UseCount, ale není dostupný v grafickém rozhraní. Z jeho názvu by však mělo být patrné, že se jedná o počet použití pole v souboru se šablonou.

4 Popis spouštění aplikace

Výsledná spustitelná aplikace rpt_diff.exe vyžaduje pro správné spuštění mít nainstalovaný nástroj pro porovnání dvou XML souborů, dále pak Visual Studio a v něm nainstalované rozhraní Crystal Reports for Visual Studio. Aplikace se spouští z příkazového řádku a musí mít specifikované tři nebo čtyři parametry spuštění.

Prvním povinným parametrem je úplná cesta ke spustitelnému souboru porovnávací diff aplikace, druhým povinným parametrem je cesta k souboru se šablonou a třetí parametr je volitelný a pokud je zadán, tak se očekává cesta ke druhému souboru se šablonou, který se bude s prvním souborem porovnávat. Posledním parametrem se určuje, který z objektových modelů pro převod do XML použít. Pokud se jako poslední parametr při spuštění aplikace zadá číslo nula tak bude převod prováděn pomocí modelu ReportDocument a pokud se zadá číslo jedna tak bude proces převodu probíhat podle modelu ReportClientDocument. Je tedy doporučeno spouštět aplikaci vždy s posledním parametrem 1, aby byl převod do XML kompletní a obsahoval veškeré aspekty souboru se šablonou.

Pro účely pouhého vyzkoušení XML převodu je tedy dostupná varianta, kdy se zadá jako první parametr cesta ke spustitelnému souboru aplikace, která dokáže zobrazit XML soubor, jako druhý parametr se zadá cesta k souboru se šablonou a jako poslední parametr číslo 0 nebo 1 a výsledkem bude otevření vyexportovaného XML souboru v zadané aplikaci. V případě chybného zadání parametrů je vypsána chyba a vypíše se i nápověda k použití.

4.1 Výstup aplikace

Aplikace vypisuje do konsole na standartní výstup informace o průběhu převodu. Pokud v průběhu zpracování nastane chyba je program ukončen a na standartní chybový výstup je vypsána chybová hláška a je vrácen příslušný chybový kód. V případě, že jde o chybu v zdání parametrů, tak se vypíše stručná nápověda k použití.

Možné návratové kódy:

- 0 – vše v pořádku
- 1 – špatně zadaná cesta k porovnávací aplikaci
- 2 – špatně zadaná cesta k souborům se šablonou
- 3 – špatně zadané argumenty
- 4 – špatně zvolený model pro převod do XML
- 5 – chyba převodu do XML

Soubory XML, které se při zpracování vytvoří ve stejném adresáři jako jejich zdrojové předlohy jsou po skončení aplikace odstraněny. Pro jejich ponechání je nutné v porovnávací aplikaci je uložit do jiného umístění nebo před uzavřením aplikace je překopírovat do jiného umístění na disku.

4.2 Napojení na TortoiseSVN

V TortoiseSVN lze snadno nastavit pro porovnání dvou verzí souboru vlastní příkaz, který má být proveden. Nastavuje se zvlášť pro každou příponu souborů. Pro naše použití, je tedy vhodné nastavit pro přípony .rpt spuštění aplikace rpt_diff a jako její parametry nastavit:

1. cestu ke spustitelnému souboru porovnávacího software
2. %base
3. %mine.
4. číslo specifikující model pro převod

Což znamená, že se po vyžádání porovnání dvou verzí souboru se šablonou přes rozhraní TortoiseSVN otevře porovnávací software, kde vlevo bude původní verze a vpravo bude nová verze souboru.

4.3 Možná rozšíření

Mezi vhodná rozšíření se řadí například možnost sloučení změn (*merge*), XSLT transformace XML souboru pro lepší přehlednost nebo optimalizace XML značek, aby se například prázdné značky vynechaly a pro jednodušší reporty, které neobsahují veškeré dostupné části, by se snížila velikost výsledného XML souboru a tím i jeho přehlednost. Nebo případný převod XML do jiné textové reprezentace jako například YAML nebo JSON.

Důležitou částí pro možnost slučovat změny je naimplementovat opačný převod tedy z XML zpět do rpt, tím by se umožnilo provádět i jiné změny v XML, které by se daly promítnout do souboru se šablonou.

Dalším vhodným rozšířením by mohla být implementace vlastního nástroje na grafické procházení změn.

5 Existující nástroje

Crystal Reports v sobě obsahuje možnost export do XML tato možnost však obsahuje pouze grafický vzhled výsledného zobrazení a nelze tak z tohoto XML exportu získat informace o podrobných nastaveních a formulích. Takový export se tedy spíše hodí pro vytvoření XML souboru na základě šablony, podobně jako je možné vytvořit PDF, XLS, DOC a podobné formáty výsledného zobrazení dat, ale pro porovnání dvou verzí souboru se šablonou to vhodné není.

Nenašel jsem existující externí nástroj, který by přesně odpovídal tomuto řešení, tedy bezztrátovému převodu souboru se šablonou do XML reprezentace. Nejvíce podobný byl nástroj RPTtoXML, který provádí převod do XML pomocí modelu ReportDocument, a tedy neobsahuje veškerou funkcionalitu a je stále ve vývoji a je dostupný na GitHubu[2].

Dále jsem objevil nástroj RptDiff od Retsel Group. Tato společnost vyvíjí produkt nazvaný ReportMiner, který je schopen procházet soubor se šablonou a poskytuje analýzu reportu a jeho subreportů, detailní analýzu rozdílů v reportech možnost přichystat si návrh šablony ve Wordu pomocí parametrů a je schopen vytvořit databázi atributů souborů se šablonami pro analýzu více souborů najednou. Jako další produkt nabízí samostatný RptDiff, který je pouze doplňkem do Microsoft Visual SourceSafe. Bohužel ve zkušební verzi, která je dostupná zdarma, není tato funkce plně podporována, a proto nemohu objektivně posoudit, do jaké míry je porovnání dvou šablon detailní.

Aplikace R-Tag obsahuje vlastní verzovací nástroj, který dokáže udržovat jednotlivé verze souborů se šablonou a současně velmi pěkné grafické znázornění rozdílů v jednotlivých verzích reportu, a i spoustu dalších operací jako vyhledávání s různými možnostmi nastavení nebo vytvoření dokumentace na základě analýzy souborů se šablonou. Opět se stejně jako v případě ReportMiner jedná pouze o placený nástroj, a tak jsem jej nevyzkoušel, pouze jsem viděl propagační video.

Ostatní nástroje na analýzu reportů, které jsem našel, jako například Report Analyzer od Cortex systems nebo Report Runner Documentor od Jeff-Net, již neposkytovaly možnost porovnávání dvou souborů se šablonou.

[2] https://github.com/ajryan/RptToXml

6 Testování

Pro otestování funkčnosti a představení výsledků programu si nejprve vytvořím jednoduchý soubor se šablonou, ve kterém nebudou využity uživatelské formule. Pro tento soubor poté porovnám XML výstup při převodu pomocí modelů ReportDocument a ReportClientDocument. V tomto prvním případě by měl být výhodnější model ReportDocument.

Poté provedu druhý test, kde soubor se šablonou bude obsahovat většinu možných nastavení a výsledné XML soubory po převodu za pomoci obou objektových modelů se budou lišit v množství obsažených informací již více. Ukáže se zde síla při zpracování pomocí modelu ReportClientDocument.

Nakonec provedu test v podobě reálného použití. Nastavím si konsolovou aplikaci do TortoiseSVN a zobrazím porovnání. Zhodnotím, zda změny provedené v souboru se šablonou jsou lehce dohledatelné z následného porovnání vyexportovaných XML souborů v aplikaci KDiff.

6.1 Jednoduchý report

Pro první případ jsem využil jednoduchý soubor se šablonou EasyRpt1.rpt, který je dostupný v příloze a jeho lehce upravená verze EasyRpt2.rpt je taktéž dostupná jako příloha. Tento report neobsahuje žádné napojení do databáze a obsahuje pouze 7 textových polí s různým nastavením fontu a pár grafických rámečků. Rozdíl mezi prvním a druhým rpt souborem je pouze v jednom z textových polí a jde o změnu textu. Rozdíly v těchto dvou souborech po převedení do XML pomocí obou možných objektových modelů jsou snadno dohledatelné, a to tedy dokazuje, že využití obou modelů je užitečné.

Při použití objektového modelu ReportDocument má výsledný XML soubor 120 řádků a oproti převodu pomocí druhého objektového modelu ReportClientDocument je tedy menší, protože při převodu pomocí něj má výsledný XML soubor řádků 218. Avšak i pouze u takto jednoduché změny je vidět, že využití druhého z modelů poskytuje více možností. Je zde navíc informace o verzi souboru se šablonou a lze tedy říci, který ze souborů byl původní a který je nový.

6.2 Složitý report

Pro druhou část testování jsem vytvořil soubor se šablonou ComplexRpt1.rpt, který je dostupný jako příloha a následně jsem jej upravil na více místech pro demonstrování rozsáhlejších úprav v souboru a uložil jako soubor ComplexRpt2.rpt, který lze taktéž dohledat v přílohách. Report obsahuje všemožné nastavení od připojené databáze přes nastavení políček či nastavení seskupení až po detaily jako třeba nastavení barev v závislosti na uživatelských formulích. Změnu jsem záměrně udělal netypicky téměř ve všech nastaveních. Obvykle se totiž při reálném využití v rámci jedné změny šablony neprovádí více úprav najednou.

Po využití aplikace rpt_diff na takto rozdílné soubory je překvapující, že porovnání je stále dosti čitelné zejména díky XML struktuře, která i v případě, že některá část v jednom ze souborů chybí je jasně znázorněna prázdným místem a lze tedy snadno zjistit, které části byly přesně odstraněny a které pouze změněny. V tomto případě je změněných řádků mnohem více, než v předchozím testu což se také promítá na počtu řádků jednotlivých XML výstupech. Při použití modelu ReportClientDocument má XML výstup pro report ComplexRpt1 1054 řádků a pro druhý report jen 839 řádků. Již z tohoto jde vidět, že druhý report je osekanou verzí toho prvního. A také jako jeden z prvních rozdílů v souborech lze vidět změna datového modelu, kde jsou odstraněný dvě tabulky a následuje spousta dalších změn.

Při použití modelu ReportDocument jsou výsledné XML soubory přibližně o 100 řádků kratší což odpovídá chybějícím informacím, které za pomocí tohoto modelu není možné získat. Velikost XML souborů na disku při použití modelu ReportDocument je oproti velikosti XML souborů s použitím modelu ReportClientDocument téměř poloviční a díky tomu je také převod již znatelně rychlejší v řádu několika vteřin.

6.3 Reálné využití

Před možností reálně vyzkoušet aplikaci v provozu je nutné nastavit pracovní stanici. Základním předpokladem je nainstalované Visual Studio ve verzi minimálně 2010 či novější. Poté podle kapitoly 2.4 je nutné nainstalovat programové rozhraní Crystal Reports pro Visual Studio. Dalším nutným programovým vybavením je aplikace schopná porovnat dva XML soubory. Jak popisuji v kapitole 2.2, zvolil jsem si aplikaci KDiff a pro ukládání verzí souborů se šablonou využívám subversion, konkrétně grafické rozhraní TortoiseSVN. Do tohoto grafického rozhraní podle návodu v kapitole 4.2 jsem nastavil pro soubory s příponou .rpt následující příkaz:

```
"C:\rpt_diff\bin\Release\rpt_diff.exe" "C:\Program Files\KDiff3\kdiff3.exe" %base %mine 1
```

Nyní již stačí spustit porovnání dvou rpt souborů nebo porovnání verze souboru v historii přes grafické rozhraní TortoiseSVN a výsledkem bude zobrazení porovnání dvou XML souborů.

7 Závěr

Cílem této bakalářské práce bylo vytvoření aplikace pro porovnání tiskových šablon vytvořených pomocí Crystal Reports. Po prostudování dostupných nástrojů jsem zjistil, že neexistuje žádný volně dostupný nástroj s takovouto funkcionalitou. Přišel jsem pouze na komplexní nástroje s mnohem většími možnostmi na úpravu šablon či na vytváření dokumentace k šablonám.

Dále jsem prostudoval možnosti, jakými je možné získat informace ze souborů se šablonou. V této práci jsem popsal dostupné rozhraní Crystal Reports for Visual Studio a jeho jednotlivé objektové modely s jejich možnostmi využití a vybral si vhodný objektový model pro implementaci řešení zadaného problému. Pro možnost objektivního posouzení užitečnosti vybraného modelu jsem zvolil dva nejvíce vyhovující, a to ReportDocument object model a ReportClientDocument object model. Navrhnul jsem a následně implementoval aplikaci, která může využívat oba tyto modely pro převod souboru se šablonou do textové podoby. Dále jsem také prostudoval možnosti zápisu textových informací do určité struktury přehledné jak pro strojové zpracování, tak pro čtení člověkem. Nejvíce vyhovujícím formátem byl značkovací jazyk XML.

Po dokončení implementace, v této zprávě uvádím možnosti, jak s vytvořeným nástrojem pracovat a také možnost jeho napojení na grafické rozhraní pro subversion, tedy TortoiseSVN. Pro správné fungování aplikace rpt_diff, což je název právě vytvořeného nástroje pro porovnávání reportů, je nutné mít nainstalované Visual Studio a rozhraní Crystal Report for Visual Studio a dále je potřeba aplikace schopná porovnat dva XML soubory, kde jako příklad uvádím aplikaci KDiff.

V neposlední řadě jsem navrhnul pár vhodných rozšíření této práce, mezi ty nejpřínosnější z nich bych zařadil možnost opačného převodu z XML struktury zpět do souboru se šablonou. Dle mého názoru by pro implementaci takovéhoto rozšíření mělo stačit rozhraní Crystal Reports for Visual Studio za využití objektového modelu ReportClientDocument a bylo by velkým přínosem pro možnost slučování dvou změn v reportu.

Na závěr jsem provedl několik testů pro ověření funkčnosti a využitelnosti vytvořeného nástroje a také pro zhodnocení přehlednosti textové struktury v jazyce XML. Jednotlivé testy jsou také popsány v této práci. Soubory využité při testování jsou dostupné na přiloženém paměťovém médiu.

Literatura

[1] *Crystal Reports* [online]. Walldorf: SAP SE, 2018 [cit. 2018-05-13]. Dostupné z: https://www.crystalreports.com/

[2] KOSEK, Jiří. *XML pro každého*. Praha: Grada Publishing, spol., 2000. ISBN 80-7169-860-1.

[3] *Crystal Reports: Object Models for Crystal Reports for Visual Studio .NET* [online]. Crystal Decisions, 2002 [cit. 2018-05-13]. Dostupné z: https://archive.sap.com/kmuuid2/50dfb2d3-a61d-2b10-b288-a00b2e964601/crnet_object_models.pdf

[4] *TortoiseSVN* [online]. The TortoiseSVN team, 2018 [cit. 2018-05-13]. Dostupné z: https://tortoisesvn.net/

[5] *SAP Crystal Solutions Licensing* [online]. Walldorf: SAP SE, 2017 [cit. 2018-05-13]. Dostupné z: https://www.sap.com/documents/2015/07/368edae4-597c-0010-82c7-eda71af511fa.html

[6] *Crystal Reports Tutorial* [online]. 2018 [cit. 2018-05-13]. Dostupné z: https://www.tutorialspoint.com/crystal_reports/index.htm

[7] *JSON* [online]. [cit. 2018-05-13]. Dostupné z: http://www.json.org/

[8] *What is difference between JSON, XML and YAML* [online]. William Wong, 2015 [cit. 2018-05-14]. Dostupné z: http://www.electronicdesign.com/dev-tools/whats-difference-between-json-xml-and-yaml

[9] *YAML* [online]. 2011 [cit. 2018-05-14]. Dostupné z: http://yaml.org/

Seznam obrázků

Přílohy

Příloha A

Obsah CD

- /doc – technická zpráva
- /src – zdrojové soubory aplikace
- /test – soubory rpt z testování
- /models – objektové modely

Příloha B

Kompletní UML diagram